УДК 004.8

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

# АТЕСТАЦІЙНА РОБОТА
# Пояснювальна записка

Рівень вищої освіти другий (магістерський)

ГЮИК.502800.005 ПЗ

Дослідження методів організації штучного інтелекту в ігровому процесі

(тема)

Виконав:
Студент 2 курсу, групи *ІТПм-19-1*
Спеціальність *122 – Комп'ютерні науки*
(код і повна назва напряму)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційні технології проектування*
(повна назва освітньої програми)

*Задорожна І.О.*
(прізвище, ініціали)

Керівник *Губаренко Є.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Гребеннік І. В.*
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 – Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційні технології проектування*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20___ р.

# ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові *Задорожної Ірини Олександрівни*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Дослідження методів організації штучного інтелекту в ігровому процесі»*

затверджена наказом по університету від *« 02 » 11* 20*20* р. № *1517 Ст*

2. Термін подання студентом роботи (проекту) *22 грудня 2020 р.*

3. Вихідні дані до роботи (проекту) *Середовище, на якому відбувається взаємодія між головним героєм та супротивником. Латаття, по якому пересуваються головний герой та супротивник. Головний герой, ціль якого обійти супротивника та дійти до кінця. Супротивник, ціль якого не дати головному герою дійти до кінця гри.*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) *Вступ. Аналіз предметної області. Виявлення проблем та актуалізація рішення. Опис та порівняння аналогів. Постановка задачі. Навчання з підкріпленням. Алгоритм Q-learning. Алгоритм SARSA. Порівняння on-policy та off-policy алгоритмів. Вхідні дані експерименту. Опис експерименту. Порівняння результатів реалізації Q-learning та SARSA методів. Ігровий процес. Програмна реалізація. Створення дизайну гри. Висновки.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів) *Плакати на аркушах формату А4: основні позначення гри, рівень навчання, головне меню гри, платформа з бонусами, процес навчання суперника пересуватися дозволеним маршрутом, правильний маршрут агента, процес навчання супротивника з перешкодою, графік результатів протягом 3000 ігрових епізодів метода Q-learning, середнє значення за кожні 100 епізодів метода Q-learning, графік результатів протягом 3000 ігрових епізодів метода SARSA, середнє значення за кожні 100 епізодів метода SARSA, порівняння Q-learning та SARSA алгоритмів.*

6. Консультанти розділів роботи (проекту)

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| *1* | *Отримання, аналіз завдання, уточнення* | *02.11.2020* | |
| | *плану роботи* | | |
| *2* | *Аналіз предметної області* | *05.11.2020* | |
| *3* | *Постановка задачі* | *10.11.2020* | |
| *4* | *Проведення експериментальних досліджень* | *25.11.2020* | |
| *5* | *Оформлення пояснювальної записки* | *01.12.2020* | |
| *6* | *Підготовка презентації* | *09.12.2020* | |
| *7* | *Подання закінченої роботи науковому керівникові* | *10.12.2020* | |
| *8* | *Подання роботи на рецензування* | *11.12.2020* | |
| *9* | *Попередній захист* | *15.12.2020* | |
| *10* | *Подання роботи до екзаменаційної комісії* | *22.12.2020* | |

Дата видачі завдання *02 листопада 2020 р.*

Студент _____ *Задорожна І.О.*
(підпис)

Керівник роботи _____ *доц. Губаренко Є.В.*
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка до атестаційної роботи: 76 с., 1 табл., 34 рис., 3 додатків, 24 джерела інформації.

ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ГРА, GAMEDEV, ІГРОВА ІНДУСТРІЯ, SARSA, Q-LEARNING, ON-POLICY, OFF-POLICY

Об’єкт дослідження – штучний інтелект управління ігровим процесом.

Предмет дослідження – алгоритм навчання з підкріпленням SARSA та Q-learning.

Мета атестаційної роботи є дослідження та порівняння результатів навчання агента за допомогою алгоритмів SARSA та Q-learning.

Методи досліджень – методи системного аналізу і синтезу, прогнозування та експертних оцінок, багатофакторний аналіз, машинне навчанні з підкріпленням.

Результат атестаційної роботи – розроблено рекомендації до вимог застосування алгоритмів машинне навчанні з підкріпленням SARSA та Q-learning.

Галузь застосування – результати можуть бути використані у процесі розробки штучного інтелекту керування ігровим процесом та негровими об’єктами.

# ABSTRACT

Master’s explanatory note: 76 pages, 1 tables, 34 figures, 3 applications, 24 sources.

ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, REINFORCEMENT LEARNING, GAME, GAMEDEV, GAME INDUSTRY, SARSA, OFF-POLICY, ON-POLICY

The object of research is the artificial intelligence of game control.

The subject of the research is a learning algorithm with reinforcement of SARSA and Q-learning.

The purpose of the certification work is to study and compare the results of the agent's training using SARSA and Q-learning algorithms.

Research methods – methods of system analysis and synthesis, forecasting and expert assessments, multifactor analysis, machine learning with reinforcement.

The result of the certification work – developed recommendations for the requirements of the application of machine learning algorithms with reinforcement SARSA and Q-learning.

Field of application – the results can be used in the development of artificial intelligence to control the gameplay and non-gaming objects.

ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПК – персональний комп’ютер;

ШІ – штучний інтелект;

МН – машинне навчання;

RL – навчання з підкріпленням;

Gamedev – це процес розробки гри під певну ігрову платформу;

DARPA – це агентство передових оборонних дослідницьких проектів;

MDP – процес прийняття рішення Маркова;

VR – Virtual Reality – віртуальна реальність;

AR – Augmented Reality – доповнена реальність;

UX – User Experience – досвід користувача;

UI – User Interface – інтерфейс користувача;

SARSA – середовище, дія, нагорода, середовище, дія.

# ВСТУП

Історія розвитку штучного інтелекту починалась з 1956 року, але лише сьогодні через удосконалення засобів зберігання даних, збільшення обсягів даних, удосконалення алгоритмів, оптимізації обчислюваних потужностей він став неймовірно популярним [1].

В 50-х роках минулого століття проводились перші дослідження ШІ для систем символьних обчислень і вирішення проблем з ними. Наступним кроком було навчання комп'ютерів імітувати розумову діяльність людини. В 70-х роках саме агентство передових оборонних дослідницьких проектів (DARPA), яке відповідає за розробку нових технологій для використання в збройних силах США намагалося виконати проект по створенню віртуальних вуличних карт. І в 2013 році фахівці DARPA створили першого інтелектуального особистого помічника.

Ці роботи лягли за основу для формальної логіки міркувань і принципів автоматизації, які зараз використовуються в системах, які покликані примножувати і доповнювати можливості людини, а саме в розумних пошукових системах і системах для підтримки прийняття рішень.

Хоча штучний інтелект в багатьох науково-фантастичних романах і фільмах зображується як людиноподібний робот, але на даному етапі нашого життя технології ШІ далеко не такі розумні і зовсім не страшні.

В даній роботі розглядається одна з областей штучного інтелекту – машинне навчання. Вона надає можливості комп'ютеру “вчитися”, використовуючи надані дані, поступово покращувати якість виконання роботи конкретної задачі, не будучи попередньо запрограмованим [2].

МН досліджує вивчення і побудову алгоритмів, які можуть навчатися робити прогнози на даних, використовується в ряді обчислювальних задач,

де є неможливим проектування і програмування явних алгоритмів з хорошою продуктивністю.

Наразі машинне навчання набуває нових форм і постійно розвивається. Воно будується на концепції, що машини здатні самостійно адаптуватися і постійно вчитися. Комп'ютери вчаться на попередніх показниках і обчисленнях, щоб отримати надійні і успішні рішення для створення кращого кінцевого результату. Тобто вони можуть робити те, на що не були запрограмовані спочатку.

Майбутнє технологій за машинним навчанням. Те, що сьогодні робиться вручну людиною, завтра будуть робити машини. Алгоритми машинного навчання не тільки будуть використані в економіці та бізнесі, а й міцно увійдуть в щоденне використання.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Аналіз предметної області

Індустрія створення відеоігор сьогодні являється найбільшою індустрією з поміж усіх існуючих, вона більше ніж кіноіндустрія та музична (рис. 1.1).


Рисунок 1.1 – Місце ігор у повсякденному житті

Це не дивно, адже саме вона культивує у собі безліч різних сфер і надає користувачеві найбільше враження зануреності у дію, що розгортається перед гравцем. Для відео ігр створюють музику, яку грають справжні оркестри, пишуть сценарії, які часом кращі, ніж для кіно, малюють неймовірні зображення та створюють тривимірні моделі та світи, які з кожним роком стають все детальними, більш живими та справжніми. З кожним роком ігри все дедалі еволюціонують.

Мобільний геймінг займає друге місце за популярністю після комп’ютерів і на сьогодні це найбільш відкрита платформа для всіх, хто хоче спробувати свої сили у gamedev [3]. Але кожного дня на платформах «App Store» та «Google Play» виходить по 200, або навіть більше ігр, що робить ці платформи майже недоступними для звичайних користувачів з низьким маркетинговым бюджетом.

Найбільшої популярності набули ігри з штучним інтелектом. Для додання почуття реальності в ігри поміщають різного роду ШІ (рис 1.2). Усі IT-сфери зробили величезний стрибок у розвитку, та ж сама тенденція спостерігається і в ігровій індустрії [4].

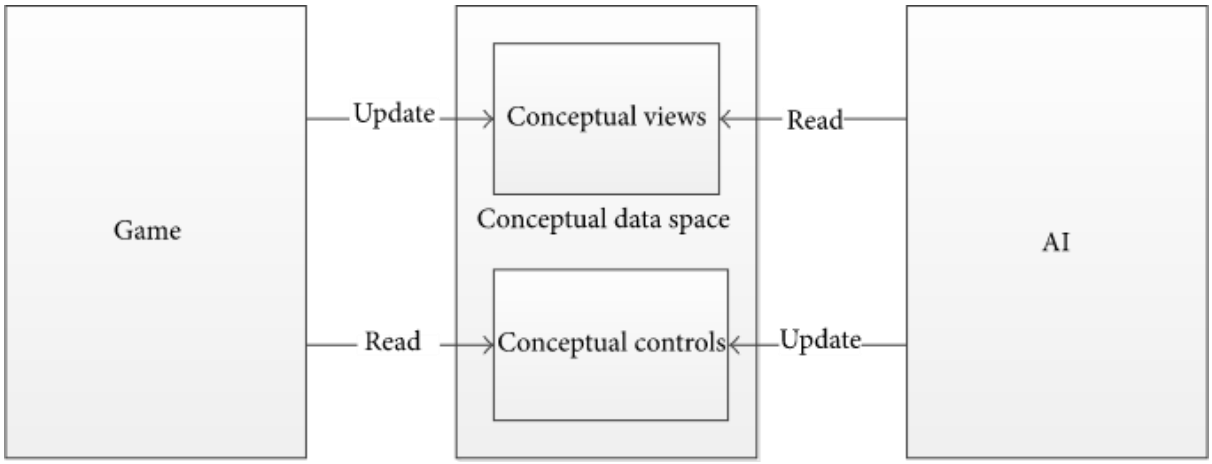


Рисунок 1.2 – Основна архітектура відеоігри з використанням концептуального ШІ

Технології штучного інтелекту є невід'ємною частиною індустрії комп'ютерних ігор. З моменту появи першої гри користувачам завжди хотілося бачити в особі комп'ютера сильного і розумного суперника (рис.1.3).

Вирішити таку задачу можна за рахунок наділення ігрових комп'ютерних агентів інтелектуальними здібностями, схожими з тими,

якими володіє людина. Іншими словами, необхідно використовувати спеціалізовані системи ігрового штучного інтелекту.

Рисунок 1.3 – Штучний інтелект в ігровому світі

Штучний інтелект відповідає за модуляцію або імітацію природного поведінки гравців або окремих об'єктів. Принцип зводиться до імітації поведінки, об'єктами керує не людина. Інакше кажучи, ШІ є штучною заміною людського інтелекту. У деяких іграх використовується найпростіший ШІ, що включає лише невеликий набір правил. У сфері ігрової індустрії ШІ має практичну роль, а не розважальну. Тут не потрібна наявність емоційності, самосвідомості, самостійної навченості. Все необхідне знаходиться в межах однієї системи, тому коло знань звужується. Головне завдання ШІ складається в правдоподібною і переконливою імітації поведінки гравців [5].

## 1.2 Виявлення проблем та актуалізація рішень

Ми занурені в епоху відродження штучного інтелекту, який безпосередньо впливає на розвиток ігрового ШІ і, як наслідок, покоління механізмів прийняття рішень представляє захоплюючий виклик для наукової спільноти, оскільки ця мета може бути здійснена з багатьох різних точок зору (рис. 1.4).

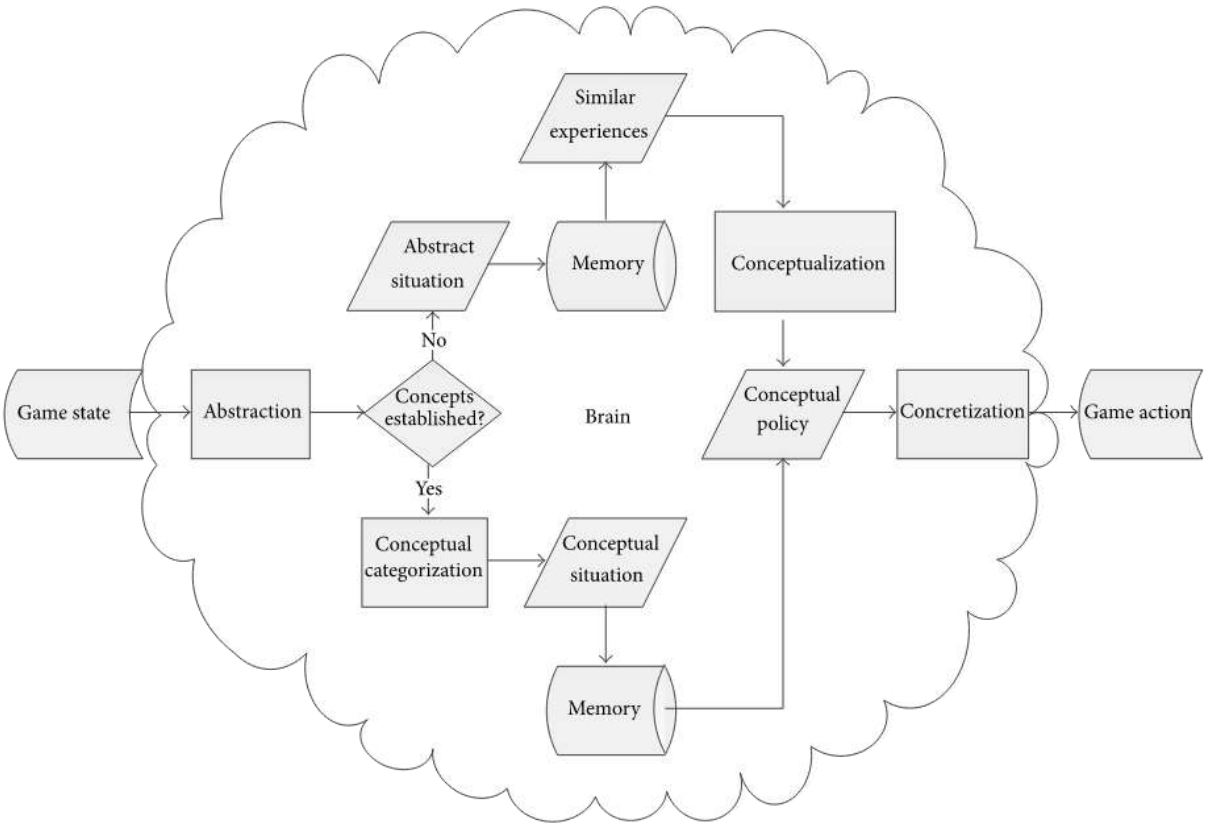


Рисунок 1.4 – Можливий процес прийняття людських рішень у відеоіграх

Вдосконалення ігрового процесу безпосередньо залежить від розробки імітації людської поведінки. Сучасні гравці вимагають найякісніших суперників, які виявляють розумну поведінку. Крім того, особливо в онлайнових іграх, загальновідомо, що гравці із задоволенням грають проти інших гравців-людей, проте також відомо, що багато гравців

– це боти, створені розробниками ігор, і це може зменшити занурення гравця в гру (рис 1.5).

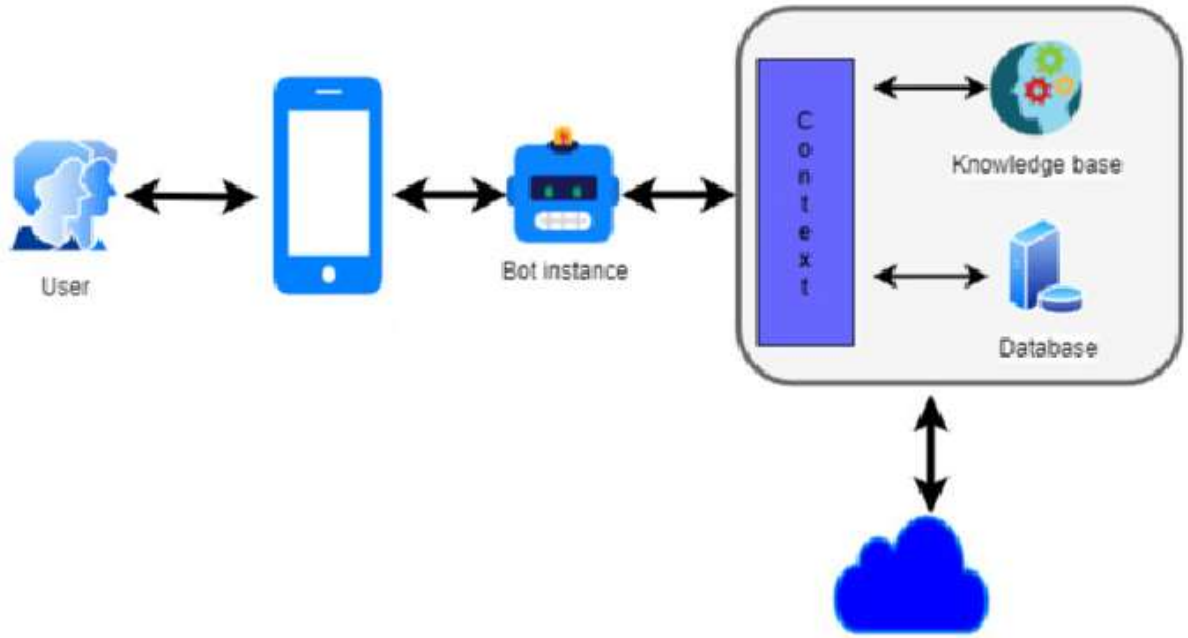


Рисунок 1.5 – Створення ботів і ігровому ШІ

Отже, приділяється багато часу для генерування ботів, які імітують гру в людському стилі з метою забезпечення відчуття того, що ви зіткнулися з іншими реальними гравцями (що насправді можуть бути не людьми) [6].

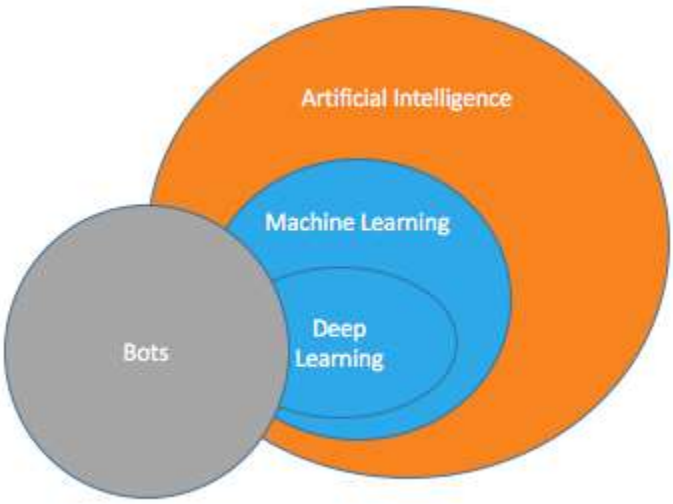


Рисунок 1.6 – Роль ботів в ШІ

Під час розробки ігор деякими розробниками розглядається підхід до створення ілюзії інтелекту, як частини ігрового ШІ. Проте, цей метод включає методики, що не можуть бути частиною рушія ігрового ШІ, саме тому цей погляд є суперечливим (рис 1.6).

Наприклад, для цього використовують інформацію про можливі зіткнення із об’єктами середовища. Ця інформація допомагає створювати ботів, що здатні уникати небажаних зіткнень з об’єктами [7].

Справжній ШІ належить до систем, які самонавчаються, спроможні приймати рішення, що базуються на довільно введених даних. Штучний інтелект у ігра, як правило, базується на евристиці та декількох правилах, отриманих емпіричним шляхом, цього вистачає, щоб надати гравцю відчуття присутності суперника та приємний геймплей.

ШІ не може змагатися на рівних з людиною в іграх, в яких дуже важливий потенціал гравця. Комп'ютерам властива нелюдська точність, швидкість та реакція, вони мають перевагу над людьми. Тим часом, коли комп'ютер має доступ до абстракції рушія, людині необхідно використовувати слух і зір, які можуть бути з обмеженнями [8].

Ігровий інтелект повинен звертатися до алгоритмів візуальної обробки, але на даний час це не є можливим для таких систем.

В таких випадках, щоб шанси були рівні застосовують обманний ШІ. Обманний ШІ за відсутності належного стратегічного мислення змушений використовувати інші переваги над гравцем такі, як більш швидке пересування або більша кількість життів, тим самим урівноважуючи ігровий процес.

## 1.3 Опис та порівняння аналогів

Гра «Q*Bert». Гра Q * Bert – аркадна гра, видана компанією Gottlieb (рис 1.7). Гра стала іконою для поп-культури, завдяки простій логікі та простій механікі. Ігрове поле складається з 28 кубиків, складених в піраміду. Використовуючи один джойстик, ви починаєте зверху і міняєте колір кубиків, стрибаючи з одного на інший. У грі поступово з'являється кілька ворогів, які перешкоджають вашому прогресу.

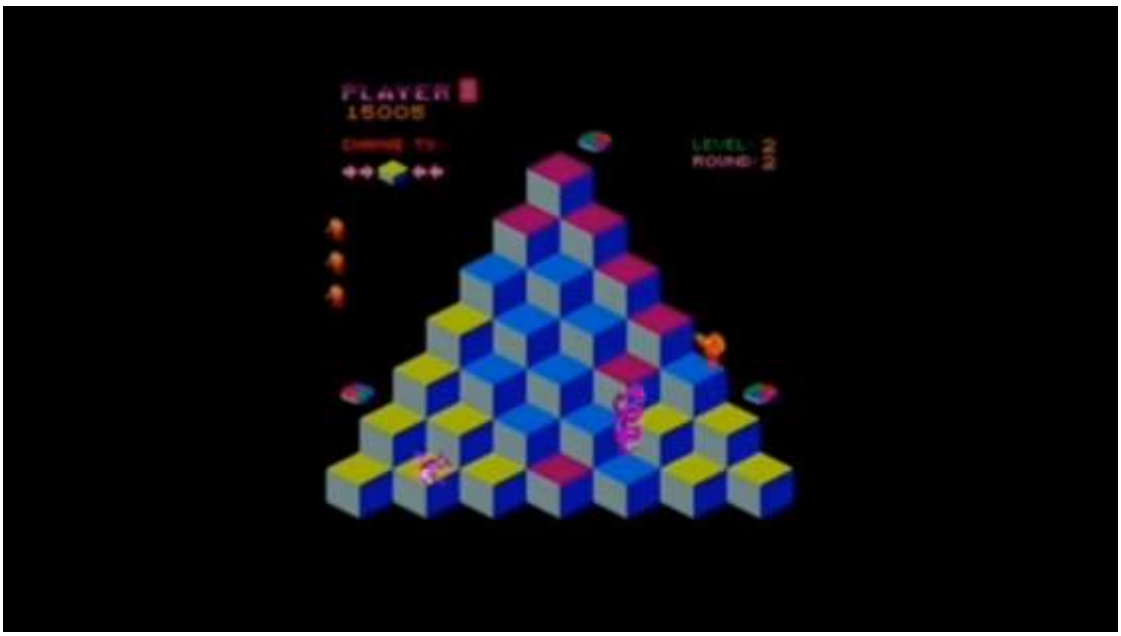


Рисунок 1.7 – Частина рівня з гри «Q*Bert»

Передбачаючи переміщення супротивників і використовуючи обхідні шляхи, гравцю потрібно поміняти колір всієї піраміди, щоб перейти до наступного рівня [9].

Гра цікава через ряд особливостей:

– гарний стиль, який динамічно змінюється;

– просте керування одним пальцем;

– офіційні рівні побудовані таким чином, що гравець стрибає під такт музики;

– низький рівень складності.

Гра «Mario». Головними героями гри є Маріо і його брат Луїджі. Мета гри – пройти через Грибне королівство, уникаючи або знищуючи солдатів черепашачого Короля купи, щоб врятувати захоплену їм у полон Принцесу. Маріо атакує супротивників, стрибаючи на них зверху або б'ючи по платформі, на якій знаходиться противник, знизу [10]. Також зустрічаються паростки, за якими Маріо піднімається на хмари, бонусний рівень, на якому велика кількість монет і немає супротивників. По дорозі Маріо збирає монети і бонуси, б'ючи по блокам зі знаком питання, а також вишукуючи секретні сховища монет в цегляних стінах. При наборі ста монет Маріо отримує додаткову «життя», спочатку у Маріо є три «життя». За переможених ворогів нараховуються очки (рис. 1.8).

Гра цікава через ряд особливостей:

– неймовірна якість аудіо ефектів;

– проста логіка гри

– багато супротивників

– багато бонусів

– наявність життів

– жодних внутрішніх ігрових покупок.


Рисунок 1.8 – Частина рівня з гри «Mario»

Гра «Crossy Road». Гра «Crossy Road» – аркадна безкінцева 3D гра, в якій гравцю необхідно стрибати, оминаючи перешкоди, потрапляти на платформи та не вийти за межі рівня (рис. 1.9).


Рисунок 1.9 – Частина рівня з гри «Crossy Road»

Камера розташована під кутом. Керування відбувається однією рукою: натискаєш – герой стрибає вперед, робиш свайп в одному із напрямків вліво, вправо чи назад – стрибаєш у цьому напрямку. Якщо декілька секунд не стрибати у напрямку «вперед» – програєш, якщо впадеш у воду, за межі рівня, чи зіткнешся з перешкодою – програєш. Кожен стрибок вперед – це 1 бал до загального рахунку під час однієї ігрової сесії. Чим більший рахунок, тим краще. За монетки, які можна отримувати кожні 6 годин, або збирати під час гри можна виграти нового персонажа[11].

Гра цікава через ряд особливостей:

– проста воксельна 3D графіка;
– просте керування;
– отримання речей через лотерею;
– велика кількість різноманітних героїв;
– короткі ігрові сесії;
– можливість грати у горизонтальній та вертикальній орієнтаціях.

Гра також дуже популярна, неодноразово компанія «Apple» застосовувала її у своїх промо матеріалах. Нараховує більше мільйона гравців. Існує для платформ iOS та Android. Перша версія була створена менше ніж за місяць.

Гра «Air Penguin». Гра «Air Penguin» – це аркадна гра з рівнями, в якій ми граємо за пінгвіна, який хоче потрапити до своєї родини (рис 1.10).

Гра має багато рівнів, навчання новим елементам та механікам відбувається поступово. Основна задача – переміщуватися вгору, але для проходженя рівнів, потрібно стрибати з платформи на платформу, а для цього необхідно маневрувати не тільки у напрямках «вперед-назад», але і «ліво-право», та уникати ворогів. При потрапляні у воду, гравець розпочинає рівень з самого початку. Керувати пінгвіном доведеться, використовуючи акселерометр мобільного пристроя [12].


Рисунок 1.10 – Частина рівня з гри «Air Penguin» та скрін магазину

Також у грі є магазин покупок, де гравець може відкрити новий вигляд для свого героя, або купити бонуси, які дозволять допустити декілька помилок під час проходження рівнів.

Гра цікава через ряд особливостей:

– приємна 2D графіка;
– керування за допомогою акселерометра пристрою;
– всі речі можна відкрити без покупок;
– короткі ігрові сесії;
– багато рівнів;
– багато різних ігрових елементів з різними механіками;
– розвиває моторику рук.

Гра була дуже популярна у 2011-2012 роках. Була на топ сходинках у магазинах AppStore та Google Play. Нараховує сотні тисяч гравців. Існує для платформ iOS та Android.

## 1.4 Порівняння аналогів

Таким чином, кожна гра має свої особливості, що зробило кожну дуже популярною й допомогло цим проектам знайти свою аудиторію фанатів, але кожна з них не ідеальна [13].

Підсумовуючи результати аналізу аналогів, можна визначити основні переваги та недоліки ігр (таблиця 1.1).

Маючи порівняння дійсно гарних аркадних мобільних проектів, з їх позитивними та негативними рисами, подивившись на функціонал, який вони пропонують окрім безпосередньо геймплея, та за рахунок яких елементів сподобалися аудиторії можна переходити до постановки задачі.

Таблиця 1.1 – Порівняльна характеристика аналогів

| Гра | Модель розповсюдження | Тривалість ігрової сесії | Особливості | Недоліки |
|---|---|---|---|---|
| Q*Bert | Преміум (1,99$) | 4 хвилини, або більше | Редактор рівнів | Висока складність |
| Mario | Преміум (4,99$) | 2 хвилини, або більше | Занурення в атмосферу гри за допомогою цікавого сюжету | Досить повільний геймплей, важко проходити рівні |
| Crossy Road | F2P (безкоштовно, з покупками та рекламою) | 2 хвилини, або більше | Велика кількість різноманітних героїв | Рандомізація відкривання нових героїв |
| Air Penguin | Преміум (0,99$) | 2 хвилини, або більше | Керування акселерометром | Мала кількість речей для відкривання і прогресу |

## 1.5 Постановка задачі

S – безліч станів середовища (гра розглядається як середовище)

Гра агента з середовищем:

- Ініціалізація стратегії $\pi 1$ (a | s) і стану середовища s1;
- Для всіх t = 1 ... T:
  - агент вибирає дію at~$\pi$t (a | st);
  - середа генерує нагороду rt + 1~p (r | at, st) і новий стан st + 1 ~ p (s | at, st);
  - агент коригує стратегію $\pi$t + 1 (a | s).

Формально найпростіша модель навчання з підкріпленням складається з:

- множина станів оточення (states) S – поточна ситуація, повернута навколишнім середовищем;
- множина дій (actions) A – всі можливі ходи, які може зробити агент;
- множина виграшів R – нагорода, яку отримує агент після певної дії.
- політика $\pi$ - стратегія, яку використовує агент для визначення наступного дії на основі поточного стану.

У довільний момент часу t агент характеризується станом st∈S і безліччю можливих дій A (st). Вибираючи дію a∈A (st), він переходить в стан st + 1 і отримує виграш rt. Грунтуючись на такій взаємодії з навколишнім середовищем, агент, який навчається з підкріпленням, повинен виробити стратегію $\pi$: S → A, яка максимізує величину R = r0 + r1 + ⋯ + rn в разі МППР, що має термінальний стан, або величину:

$$R = \Sigma t \gamma t rt,$$

для МППР без термінальних станів (де $0 \leq \gamma \leq 1$ – дисконтирующий множник для "майбутнього виграшу").

Таким чином, навчання з підкріпленням особливо добре підходить для вирішення завдань, пов'язаних з вибором між довгостроковою і короткостроковою вигодами.

Метою роботи даної передатестаційної роботи є дослідження методів навчання з підкріпленням SARSA та Q-learning на прикладі мобільної аркадної гри аби виявити переваги та недоліки наведених методів.

Функціонал, який надається програмою, повинен бути чітким і зрозумілим. Графічне відображення повинно бути сучасним, чітким, зрозумилім.

Гра, що проектується, має забезпечувати виконання наступних функціональних вимог:

- користувач має змогу заробляти досягнення;
- користувач має змогу встановлювати рекорди;
- гра повина мати графічне відображення на дії гравця;
- усі ігрові елементи повинні бути інтерактивними.

Поставлені вимогі, при їх виконанні сформують стабільну гнучку ігрову систему, яку можна буде подалі розширювати.

# 2 МЕТОДИ ОРГАНІЗАЦІЇ ШТУЧНОГО ІНТЕЛЕКТУ В ІГРОВОМУ ПРОЦЕСІ

## 2.1 Навчання з підкріпленням

Навчання з підкріпленням є галуззю машинного навчання, котра вивчає, процес прийняття рішень програмними агентами у середовищі, що має на меті максимізувати деяке поняття кумулятивної винагороди. Навчання з підкріпленням можна охарактеризувати, використовуючи поняття агентів, середовищ, станів, дій і винагород (рис 2.1).

Навчання з підкріпленням розглядається в якості однієї з трьох парадигм машинного навчання, поряд з контрольованим навчанням і неконтрольованим навчанням [14].


Рисунок 2.1 – Схема процесу навчання з підкріпленням

Цей метод відрізняється від контрольованого навчання тим, що не потребує представляти пари введення/виведення, а не оптимальні дії не мають явно коригуватися. Замість цього основна увага приділяється пошуку балансу між розвідкою невідомих стратегій і експлуатацією поточних знань.

Середовище зазвичай формується як процес прийняття рішення Маркова (MDP), так як багато реалізацій алгоритмів навчання з підкріпленням для цього контексту використовують методи динамічного програмування. Основна відмінність між класичними методами динамічного програмування і алгоритмами навчання з підкріпленням полягає в тому, що останні не передбачають знання точної математичної моделі MDP і націлені на великі MDP, де точні методи стають неможливими [15].

Навчання з підкріпленням вивчаються в багатьох інших областях, таких, як теорія ігор, теорія управління, моделювання на основі оптимізації, дослідження операцій, теорія інформації, багатоагентні системи, статистика і генетичні алгоритми. У літературі з дослідження та управління операціями навчання з підкріпленням називається наближеним динамічним програмуванням або нейродинамічний програмуванням [16].

Проблеми інтересу до навчання з підкріпленням також вивчалися в теорії оптимального управління, яка займається головним чином існуванням і характеристикою оптимальних рішень, а також алгоритмами для їх точного обчислення і, в меншій мірі, навчанням або наближенням, особливо під час відсутності математичної моделі середовища. В економіці і теорії ігор навчання з підкріпленням може використовуватися для пояснення того, як рівновага може виникати за умов обмеженої раціональності [17].

Існує агент, який взаємодіє з навколишнім середовищем, виконуючи дії. Навколишнє середовище дає нагороду за ці дії, а агент продовжує їх виконувати (рис. 2.2).

Дії – це набір всіх можливих рішень, які агент може прийняти. Дія майже не потребує пояснень, але слід зазначити, що агенти обирають серед переліку можливих дій. У відеоіграх цей список може включати біг

праворуч або ліворуч, високий або низький стрибок, присідання або стояння на місці.

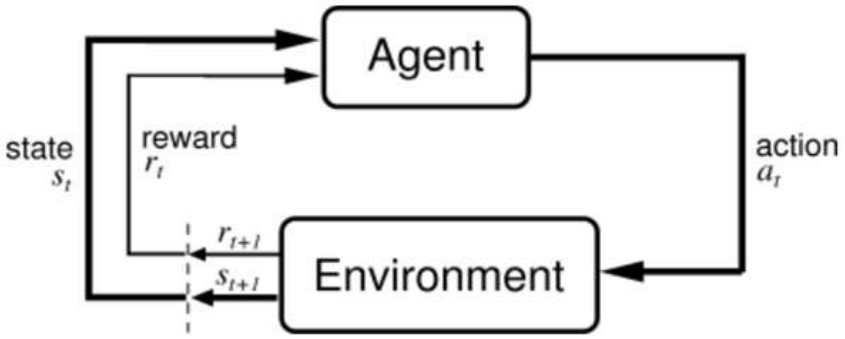


Рисунок 2.2 – Навчання з підкріпленням

Навколишнє середовище – це світ, у якому існує агент. Середовище сприймає поточний стан і дію агента як вхідні дані, і повертає агентові винагороду та його наступний стан, що став можливим у наслідку виконання дії.

Стан – це конкретна та безпосередня ситуація, в якій агент знаходиться; конкретне місце і момент, миттєва конфігурація, яка характеризує даного агента по відношенню до інших значних речей, таких як інструменти, перешкоди, вороги або призи.

Нагорода – це зворотній зв'язок, за допомогою якого вимірюється успіх, до якого призвела дія агента.

Формально найпростіша модель навчання з підкріпленням складається з:

– множина станів оточення S;

– множина дій A;

– множина винагород.

У довільний момент часу $t$ агент характеризується станом $s_t \in S$ і безліччю можливих дій $A(s_t)$. Вибираючи дію $a \in A(s_t)$, він переходить в

стан $s_{t+1}$ і отримує виграш $r_t$. Ґрунтуючись на такій взаємодії з навколишнім середовищем, агент, який навчається з підкріпленням, повинен виробити стратегію $\pi: S \Rightarrow A$, яка максимізує величину $R = r_0 + r_1 + \ldots + r_n$ що має термінальний стан, або величину: $R = \sum_t \gamma^t r_t$, (де $0 \leq \gamma \leq 1$ коефіцієнт дисконтування, який множиться на майбутні винагороди, виявлені агентом, щоб зменшити вплив цих винагород на вибір агента. Цей коефіцієнт призначений для того, щоб майбутні нагороди коштували менше, ніж безпосередні винагороди).

Поведінка – це стратегія, яку агент використовує для визначення наступної дії на основі поточного стану. Він відображає обирає дії, котрі обіцяють найвищу нагороду. Поведінка агента π(S, A) є функцією (детермінованою або стохастичною), що вказує, які дії слід приймати, знаходячись у певному стані у певний момент часу:

– детермінована поведінка – це просто відображення функції стан-дія π(S→A);

– стохастична поведінка – це розподіл ймовірностей над діями, або ймовірність всіх можливих дій, коли агент перебуває в певному стані π(A|S).

Функція винагороди $R_a(S, S')$ – це функція, яка визначає негайну винагороду, яку агент отримує після переходу зі стану до стану через дію. Функція значення $V^{\pi}(S)$ – це функція, яка визначає очікуваний прибуток, тобто загальну винагороду, яка може бути накопичена в майбутньому з поточного стану безперервного дотримання поведінки (2.1). Повернення є випадковою величиною оцінки одного епізоду, тоді як функція значення є очікуванням від усіх можливих епізодів і може починатися з будь-якого стану.

$$V^{\pi}(S) = E\left[\sum_{i=1}^{T} \gamma^{i-1} R_i\right] \quad (2.1)$$

Два елементи що роблять навчання з підкріпленням потужним інструментом: використання зразків для оптимізації продуктивності і наближення функцій для роботи у великих середовищах [18]. Завдяки цим двом ключовим компонентам навчання з підкріпленням може використовуватися у великих середовищах в наступних ситуаціях:

– модель середовища відома, але аналітичне рішення недоступно;

– дається тільки імітаційна модель середовища (предмет оптимізації на основі імітації);

– єдиний спосіб зібрати інформацію про навколишнє середовище – це взаємодіяти з нею.

Перші дві з цих проблем можна розглядати як проблеми планування (оскільки доступна деяка форма моделі), тоді як останню можна вважати реальною проблемою навчання. Однак навчання з підкріпленням перетворює обидві проблеми планування в проблеми машинного навчання.

## 2.2 Q-learning

Q-learning – це метод, що застосовується для вирішення проблеми машинного навчання при агентному підході. Агент базуючись на винагороді, що була отримана внаслідок попередніх дій у середовищі формує функціональну користь $Q$, яка допомагає йому обирати стратегію поведінки вже не випадково, а враховуючи попередній досвід взаємодії із навколишнім середовищем (рис 2.3).

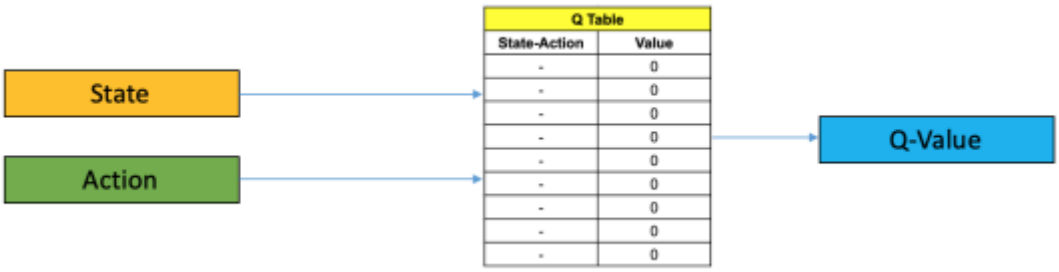


Рисунок 2.3 – Формування значення Q

Q-learning є методом навчання, що не містить моделей. Цей метод працює, вивчаючи функцію дії-значення, що в кінцевому підсумку дає очікувану корисність виконання даної дії в даному стані $s$, і притримуючись оптимальної поведінки [18]. Поведінка є правилом, за яким агент вибирає дії, враховуючи стан, в якому він знаходиться (рис. 2.4).

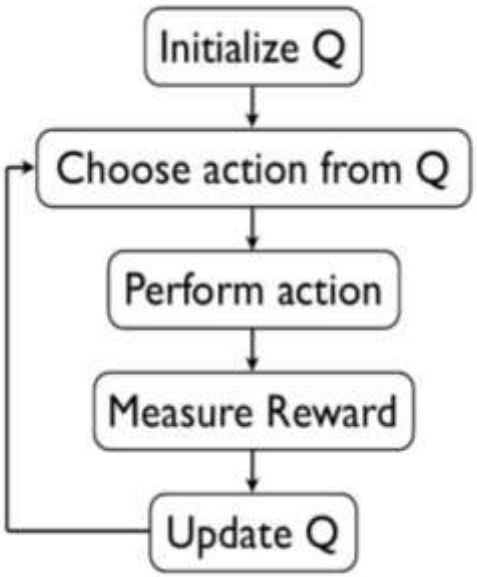


Рисунок 2.4 – Опис Q-learning

Однією із сторін що вирізняють алгоритм Q-learning серед інших є те, що цей метод здатний порівнювати очікувану корисність наявних дій, не використовуючи моделі навколишнього середовища. Крім того, Q-learning

здатний розв’язувати проблеми із стохастичними переходами та винагородами, не вимагаючи ніяких адаптацій. Q-learning як результат знаходить оптимальну поведінку, в тому сенсі, що очікуване значення загальної винагороди за всі наступні кроки, починаючи з поточного стану, є максимальними.

Вага для кроку від стану $s_t$ обчислюється як $\gamma^{s_t}$(коефіцієнт дисконту) – це число між 0 та 1 яке має ефект перебільшення винагород, що отриманні на ранніх етапах навчання, ніж отриманих пізніше (відображаючи значення «хорошого початку»). $\gamma$ також може бути інтерпретована як ймовірність успіху на кожному кроці $a_t$.

Отже, алгоритм можна зобразити функцією, яка обчислює якість комбінації дій (2.2).

$$Q: S \times A \to R. \tag{2.2}$$

До початку навчання, Q має бути ініціалізовано до довільного значення, обраного реалізацією. Після цього, у кожний момент часу $t$ агент обирає дію $a_t$, та отримує винагородою $r_t$, переходить на наступний стан $s_{t+1}$ (що може залежати як від попереднього стану $s_t$, так і від обраної дії $a_t$), тим самим уточнюючи наступне значення Q (рис. 2.5). В основу алгоритму покладено прості ітерації, що уточнюють значення використовуючи середньозважену величину попереднього значення та нову інформацію надбану після останньої дії: $Q^{new}(s_t, a_t) \leftarrow (1 - a)Q(s_t, a_t) + \alpha(r_t + \gamma max_\alpha Q(s_{t+1}, a))$, де $r_t$ є винагородою, отриманою при переході від стану $s_t$ до стану $s_{t+1}$, і α – темп навчання (0 <α<1).

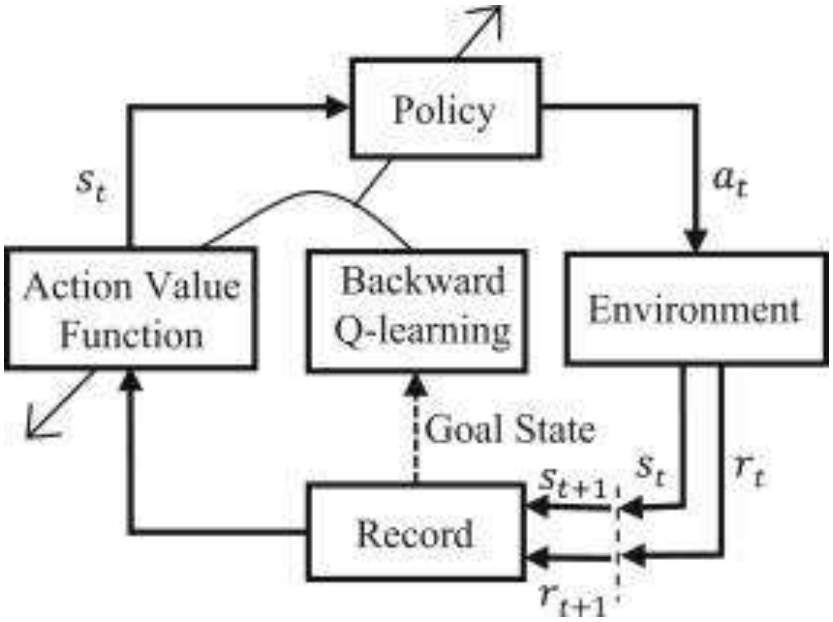


Рисунок 2.5 – Навчання методом Q-learning

Епізод навчання агента за алгоритмом завершується, як тільки досягнуто стану такого, що $s_{t+1}$ є кінцевим станом. Проте, Q-learning може також бути використаний у неепізодичних завданнях. Якщо коефіцієнт дисконту нижче 1, значення дій має бути скінченним, навіть якщо проблема може містити нескінченні петлі [19].

Для всіх скінченних станів $s_f$, $Q(s_f, \alpha)$ ніколи не оновлюється, натомість значення винагороди $r_f$ спеціально встановлюється для стану $s_f$. Зазвичай у таких випадках $Q(s_f, \alpha)$ можна вважати рівним нулю.

Швидкість або темп навчання, який також називають розміром кроку, визначає у якій мірі новий досвід може перекрити раніше отриманий досвід. Якщо фактор встановлено у 0, агент буде змушений не вивчати нову інформацію, експлуатуючи виключно досвід отриманий під час попередніх ітерацій, тоді як коефіцієнт встановлений у 1 змушуватиме агента покладатися тільки на найактуальнішу інформацію, ігноруючи попередній досвід, це може стати у нагоді для дослідження нових можливостей. У повністю детермінованих середовищах темп навчання алгоритму $\alpha_t = 1$ є оптимальним. Проте коли задача є стохастичною, при деяких технічних

умовах вимагає його зниження до 0. Але у практичних задачах зазвиай використовують постійну швидкість навчання, таку як, наприклад, $\alpha_t$= 0.1 для всіх t.

Коефіцієнт знецінення або дисконту $\gamma$ на скільки майбутні винагороди впливатимуть на результат у підсумку. Коли коефіцієнт встановлено у 0, агент буде враховувати лише найактуальніші нагороди, тобто $r_t$, у той час $\gamma$, що асимптотично наближається до 1, змушуватиме агента досягати високої нагороди у довгостроковій перспективі. Коли коефіцієнт знецінення є більшим або рівним 1, значення дій можуть розходитися. Якщо коефіцієнт дисконтування лише трохи нижче 1, навчання Q-функції призводить до поширення помилок і нестабільностей, коли функція значення апроксимується штучною нейронною мережею. У цьому випадку, починаючи з більш низького коефіцієнта дисконтування і збільшуючи його до своєї кінцевої вартості, прискорює навчання [20].

Так як Q-learning є алгоритмом, що базується на ітераціях, він неявно приймає початкову умову до того, як відбувається перший епізод навчання. Завищені значення на початку, які ще називають «оптимістичними початковими умовами», призведуть до більшої цікавості агента у дослідженнях: незалежно від того, яку дію буде обрано, правило оновлення призведе до того, що він матиме нижчі значення, ніж інша альтернатива, таким чином збільшуючи ймовірність переходу на новий стан. Початкову винагороду $r_1$ можна використовувати для скидання початкових умов. Згідно з цією ідеєю, при першому виконанні дії винагорода використовується для встановлення значення Q. Це призводить до пришвидшення навчання у випадку фіксованих винагород. Вважається, що модель, яка передбачає скидання початкових умов, прогнозуватиме поведінку агентів краще, ніж модель, яка припускає будь-яку початкову умову.

## 2.3 Алгоритм SARSA

SARSA – алгоритм, який застосовується в вирішення проблеми навчання з підкріпленням. Цей алгоритм було запропоновано Руммері та Ніранджаном у технічній примітці із назвою “Modified Connectionist Q-Learning” (MCQ-L). Назва SARSA просто відображає той факт, що основна функція оновлення Q-функції залежить від поточного стану агента $s_t$, дії, яку агент обирає $a_t$, винагороди $r_t$, яку отримує агент за вибір цієї дії, стану $s_{t+1}$, в який входить агент після здійснення цієї дії, та, нарешті, наступної дії $a_{t+2}$ , яку агент обирає виходячи зі свого нового стану. Скорочення набору $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ і дає назву SARSA (рис 2.6).

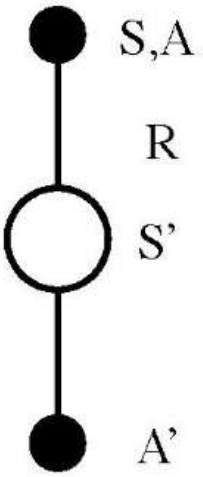


Рисунок 2.6 – Набір методу SARSA – середовище, дія, нагорода, наступне середовище, наступна дія

За алгоритмом SARSA, агент взаємодіє з середовищем та оновлює стратегію використовуючи виконані дії, отже, цей алгоритм можна віднести до класу алгоритмів навчання за стратегією (on-policy). Значення Q-функції для дії та стану оновлюється відповідно помилці, яку регулюють за

допомогою коефіцієнту темпу навчання альфа [21]. Значення Q-функції представляє сумарну винагороду, яку можна отримати за весь наступний час у межах цього епізоду, за умови виконання дії a в стані $s_{t-1}$.

Алгоритми навчання за стратегією – це алгоритми, які оцінюють та вдосконалюють ту саму стратегію, яка використовується для вибору дій. Це означає, що головна ціль такого алгоритму оцінити та вдосконалити ту саму стратегію, яку агент вже використовує для вибору дії.

Значення Q, які обчислює SARSA, залежать від поточної стратегії розвідки, тобто стратегії вибору дії $a \in A(s_t)$ яка, наприклад, може бути із випадковими кроками. Цей алгоритм може виробити іншу стратегію, ніж Q-Learning, у ситуаціях, коли за дослідження можуть бути застосовані негативні винагороди. Наприклад, коли агент підходить стану $s_t$, що може бути небезпечним для наступних етапів дослідження SARSA виявить це та обере стратегію, яка триматиме агента подалі від цього стану та знайде оптимальну стратегію, беручи до уваги важливість дослідження.

SARSA корисний при створенні агента, який досліджує у реальному часі. Для навчання у режимі офлайн, та наступного використання виробленої стратегії з агентом, що не буде виконувати дослідження, Q-learning може бути більш доречним [22].

Загалом метод обчислення Q-функції в алгоритмі SARSA можна охарактеризувати наступною формулою:

$Q^{\pi}(s,a) = E_{\pi}(\sum_{k=0}^{\infty} \quad \gamma^{k} r_{t+k+1} | s_t = s, a_t = a) = E_{\pi}(r_{t+1} +$

$\gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a)$Дії агента у середовищі:

1. Ініціалізація стратегії $\pi_1(a \mid s)$ і стану середовища $s_1$
2. Для всіх t = 1,...T,...
3. Агент обирає дію $a_t \sim \pi_t(a \mid s_t)$
4. Середа генерує винагороду $r_{t+1} \sim p(r \mid a_t, s_t)$ і $s_{t+1} \sim p(s \mid a_t, s_t)$

5. Агент робить ще одну дію $a^{|} \sim \pi_t(a \mid s_{t+1})$

6. $Q\ (s_t, a_t) = Q\ (s_t, a_t) + a_t(r_{t+1} + \gamma Q(s_{t+1}, a^{|}) - Q(s_t, a_t))$

## 2.4 Порівняння on-policy та off-policy алгоритмів

Наведені у попередніх розділах алгоритми є типовими представниками двох підгруп алгоритмів навчання з підкріпленням: on-policy та off-policy, де під “policy” мається на увазі стратегія (рис 2.7).

Алгоритм Q-learning є off-policy алгоритмом, це означає що цей алгоритм оновлює свої Q-значення, використовуючи Q-значення наступного стану $s_{t+1}$ та дію $a_{t+1}$. Іншими словами, агент оцінює Q (загальну майбутню винагороду) для стану-дії, якщо дотримуватись політики, що відмінна від поточної. У той самий час алгоритм SARSA є on-policy алгоритмом та при виборі кожної наступної дії $a_t$ завжди притримується заданої стратегії та на кожній майбутній ітерації намагається оптимізувати її для отримання найбільшої можливої кількості винагороди [23].

Основна різниця між алгоритмами, що належать до off-policy, та алгоритмами, що належать до on-policy, полягає в тому, що агентам, які навчаються за off-policy алгоритмом не потрібно дотримуватися якоїсь конкретної стратегії, агент може навіть поводитися хаотично, і, незважаючи на це, агент все ще здатний знайти оптимальну стратегію поведінки у середовищі. З іншого боку, on-policy алгоритми залежать від використовуваної стратегії. У випадку Q-learning, який є off-policy методом, агент знайде оптимальну стратегію незалежно від стратегії, яка використовується під час дослідження та отримання нової інформації, але тільки у тому випадку коли дослідження відбулося на достатній кількості станів.

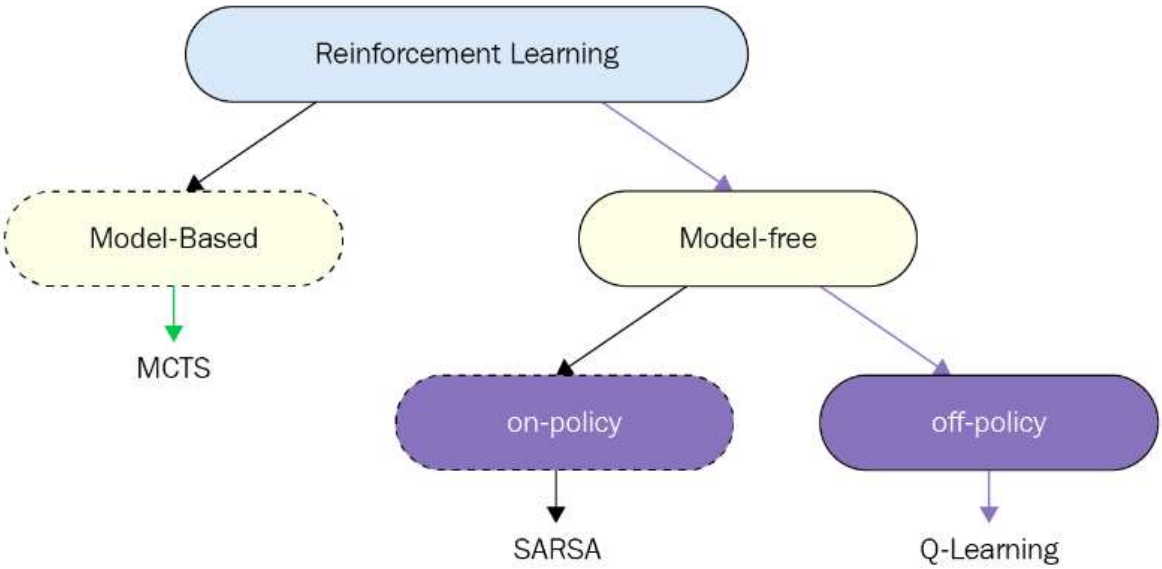


Рисунок 2.7 – Підгрупи навчання з підкріпленням

Порівнюючи Q-learning із SARSA можна сказати, що Q-learning обирає оптимальну стратегію, тоді як SARSA обирає най безпечнішу. Цей висновок полягає у тому, що існує завжди існує ризик із стратегією, що базується на дослідженні, що в будь-який момент агент Q-learning отримує велику негативну винагороду обравши дослідження. SARSA, на відміну від Q-learning, сподівається на наступну дію, щоб побачити, що насправді буде зможе робити агент після наступного кроку, і відповідно оновлює значення поточне значення Q. З цієї причини агент дізнається, що він може отримати негативну винагороду і відповідно знизить значення Q для цих пар стану-дії.

Результат полягає в тому, що Q-learning припускає, що агент дотримується найкращої з можливих стратегій, не намагаючись з’ясувати, якою ця стратегія є насправді, тоді як SARSA враховує наявну стратегію

агента та намагається її оптимізувати для отримання найбільшої винагороди у середньостроковій перспективі [24].

# 3 РОЗРОБКА МЕТОДІВ ФОРМУВАННЯ ПОВЕДІНКИ ІГРОВИХ ОБ’ЄКТІВ

## 3.1 Вхідні дані експерименту

В даній роботі була розроблена гра “Frog” для тестування алгоритмів машинного навчання, а саме Q-learning та SARSA.

Компоненти гри:

– поле, на якому розгортаються події, відбувається взаємодія між головним героєм та супротивником;

– латаття, по якому пересуваються головний герой та супротивник;

– головний герой, ціль якого пройти все поле по лататтю, не зустрітися із супротивником та дійти до кінця поля;

– супротивник, ціль якого не дати головному герою дійти до кінця і отримати нагороду;

Під час гри головний герой може пересуватися тільки по лататті, в разі порушення данного правила, гра починається з початку. Латаття може зменшуватися і зникати. Якщо головний герой не зможе уникнути супротивника, гра закінчується. Супротивник отримує нагороду, головний герой не отримує додаткові бали.

Для застосування алгоритмів Q-learning і SARSA була закладена наступна логіка:

– супротивник (агент) знає координати головного героя;

– стан середовища являє собою масив з 2 елементів, значення кожного може дорівнювати -1, 1 або 0: на даній частині поля не має героя, на даній частині знаходиться герой

Навчання супротивника (агента) відбувається в 4 етапи:

– Супротивник отримує нагороду 500 балів при зіткненні з героєм;

– Якщо герой доходить до кінця і перемагає, супротивник отримує -100 балів;

– Якщо нічого з вище вказаного не відбувається, то винагорода 10 балів;

– Значення Q встановлюється з параметрами $\gamma = 0.8$і $\alpha = 0.7$.

## 3.2 Опис експерименту

Спочатку, алгоритми були реалізовані у простий спосіб, де супротивник має рухатися тільки по лататі.

В перші хвилини навчання обидва алгоритми показали погані результати, супротивник виходив за межу дозволеного маршруту (рис 3.1).

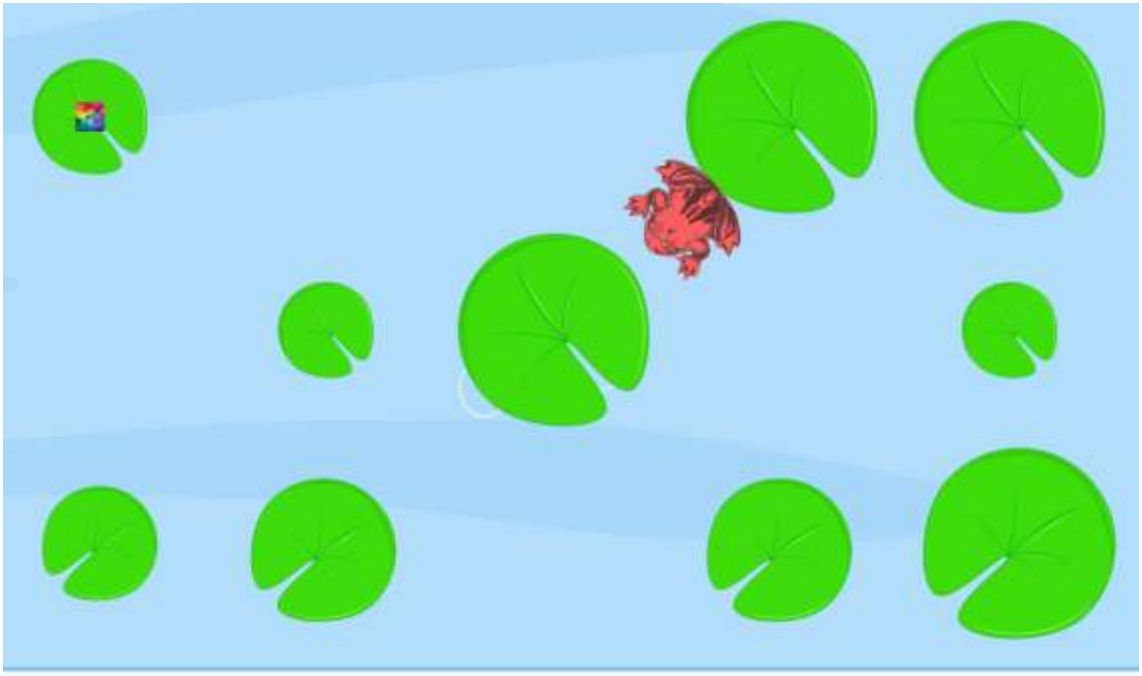

Рисунок 3.1 – Процес навчання суперника пересуватися дозволеним маршрутом

Приблизно після п’яти хвилин навчання агент рухався по правильній траєкторії (рис 3.2).

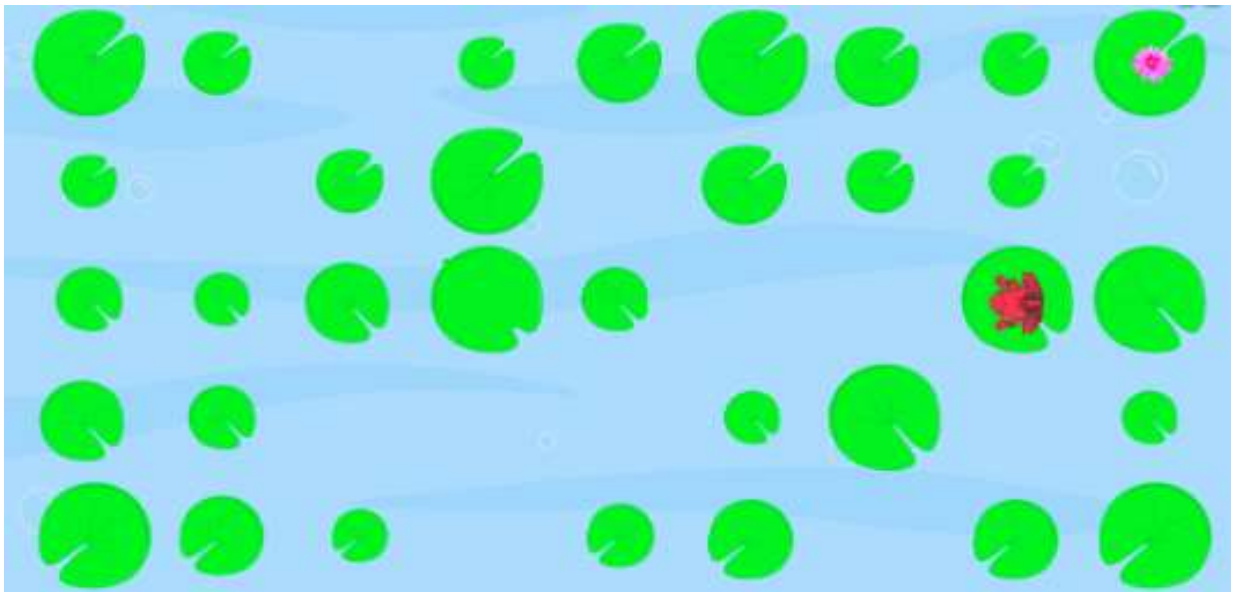

Рисунок 3.2 – Правильний маршрут агента

Далі реалізації обох алгоритмів було ускладнено таким чином, що герой не пересувається, а отже супротивник має стати перешкодою для нерухомого об’єкта. В цей раз результати були більш-менш задовільними. Після трьох хвилин навчання агент зміг дістатися головного героя (рис 3.3).

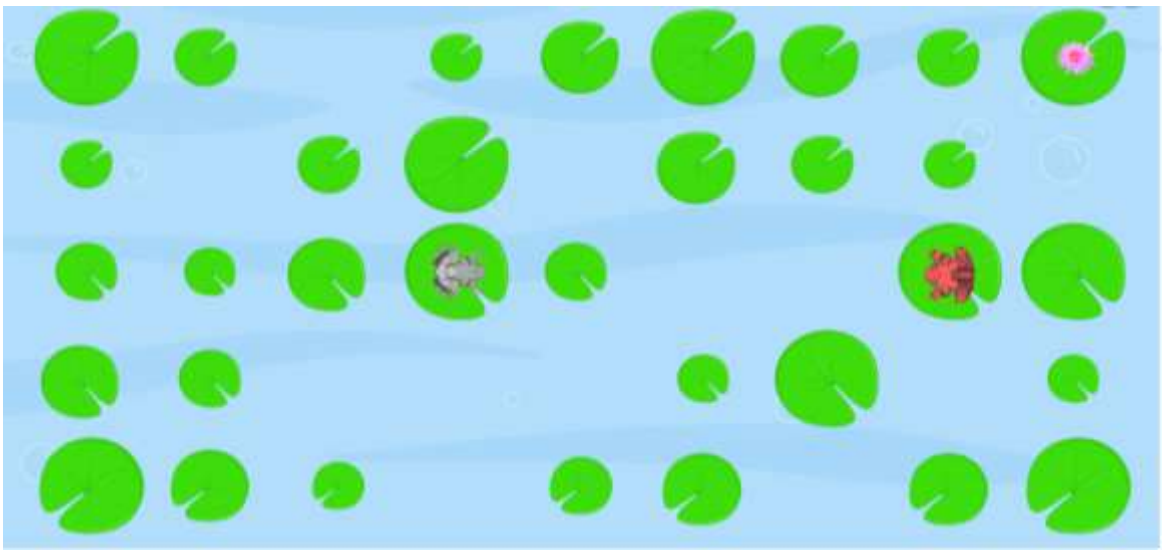

Рисунок 3.3 – Процес навчання супротивника з перешкодою

Далі було реалізовано нормальний процес змагання супротивника з головним героєм методом Q-learning.

В перші секунди навчання супротивник не рухався до головного героя, коли той змінював маршрут. Приблизно через п’ять хвилин навчання, головний герой отримував більше балів, нуж супротивник. Отже агент погано виконував свою задачу. Через десять хвилин навчання рахунок балів почав збільшуватися на користь супротивника. Максимальний рахунок 16 балів. І вже після 15-20 хвилин навчання головному герою стає важко перемогти супротивника. Максимальний рахунок 19 балів.

Наступним був реалізований метод SARSA. Умови експерименту не змінилися. В перші хвилини навчання агент навіть не рухався до своєї цілі аналогічно, як при методі Q-learning. Після 15 хвилин навчання максимальний рахунок був 18 балів, а після 20-30 хвилин 21 бал.

## 3.3 Порівняння результатів реалізації Q-learning та SARSA методів

Для отримання більш коректної картини, реалізація була ускладнена. Час тренування з 20-30 хвилин був збільшений до 2-х годин, приблизно 3000-3500 ігрових епізодів.

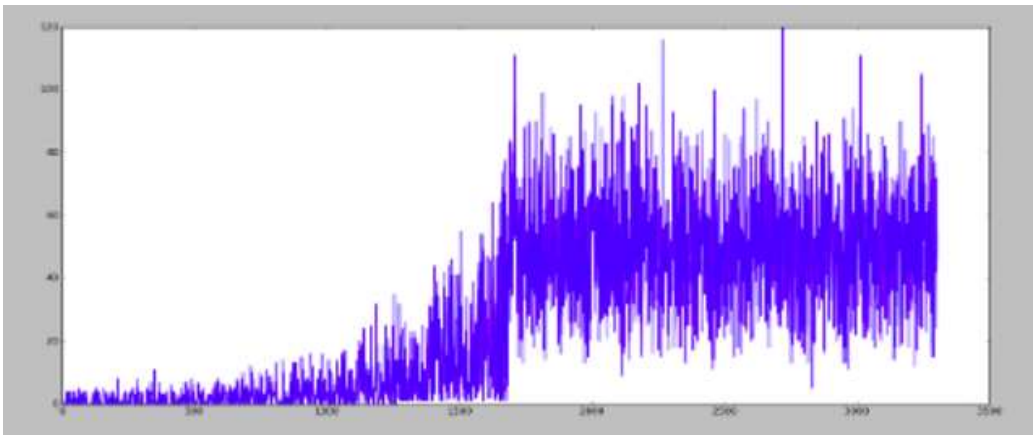

Рисунок 3.4 – Графік результатів протягом 3000 ігрових епізодів метода Q-learning

Метод Q-learning показав себе досить непогано. Приблизно після 1500 епізоду гри, показники навчання покращилися (рис 3.4). До 2000 ігрових епізодів результати навчання покращилися, але після навчання стрімко сповільнилось (рис 3.5).

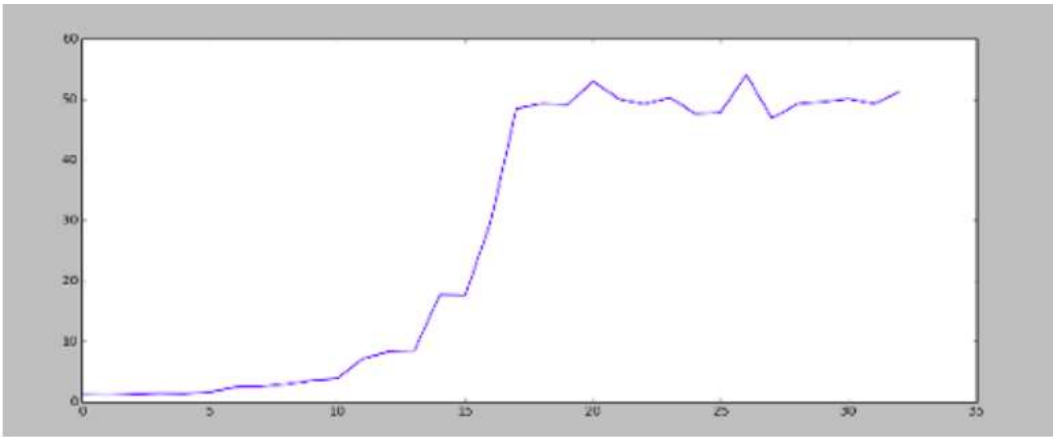

Рисунок 3.5 – Середнє значення за кожні 100 епізодів метода Q-learning

Навчання методом SARSA було виконано за тих же умов, що і попередній алгоритм. З кожним епізодом отриманий результат покращується все більше (рис 3.6).

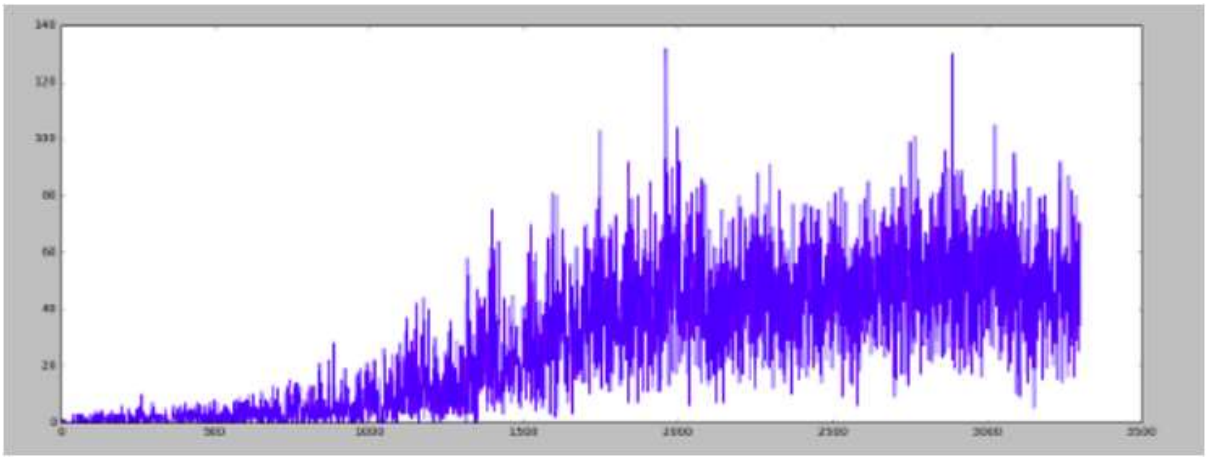

Рисунок 3.6 – Графік результатів протягом 3000 ігрових епізодів метода SARSA

Приблизно після 2500 епізоду навчання сповільнилося та середній рахунок припинив збільшуватися (рис 3.7).

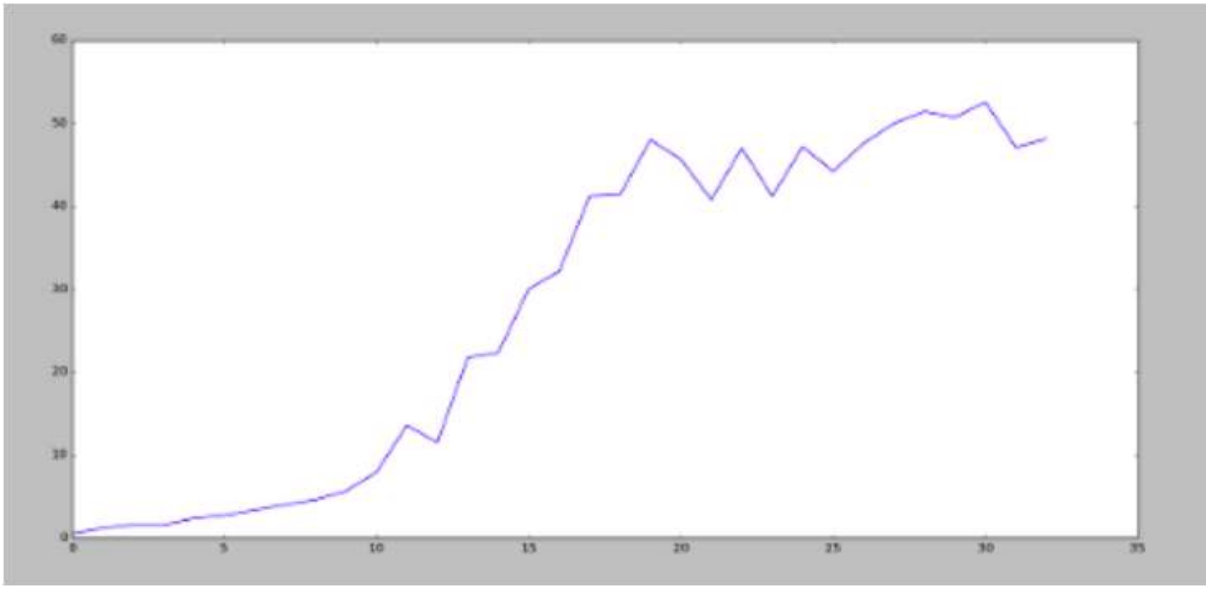

Рисунок 3.7 – Середнє значення за кожні 100 епізодів метода SARSA

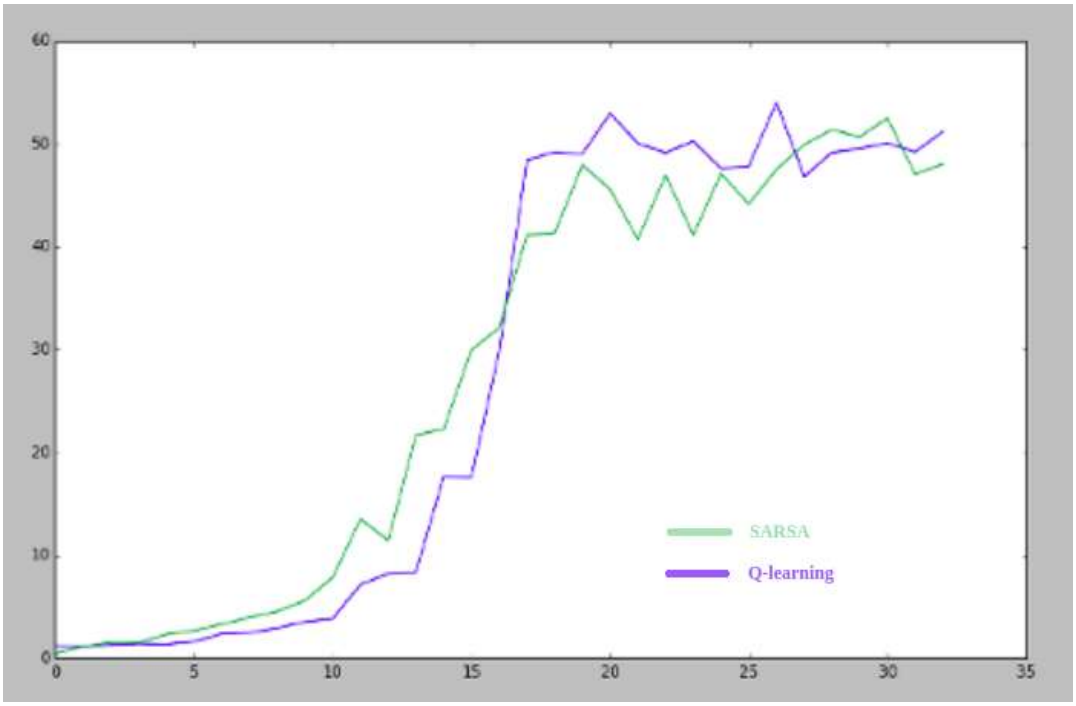


Рисунок 3.8 – Порівняння Q-learning та SARSA алгоритмів

Порівнюючи реалізацію двох алгоритмів, можна побачити, що методу SARSA потрібно більше часу навчання задля отримання задовільного результату. Метод Q-learning дає погані результати, якщо час навчання перевищенний.

На графіку видно, що обидва алгоритми показують результати на одному рівні. Але навчання методом Q-learning покращує навчання не так плавно і закінчує раніше, ніж SARSA (рис 3.8).

## 3.4 Ігровий процес

Дія відбувається на поверхні лісового ставка. Гравець починає свій шлях з будь–якої крайньої кутової позиції і його завдання перемістити в протилежний кут до квітки на лататті. Тільки латаття з квіткою і початкова не зникають з часом. У цьому вся складність завдання – пошук оптимального шляху, усвідомлення і прийняття ризику, постійна динаміка рівня, під яку повинен налаштуватися гравець. На рівні час від часу з'являються бонуси, зібравши які гравець отримує кілька очок до свого рахунку. По завершенню рівня гравець отримає бонус до рахунку в величині, що залежить від його поточного значення, що робить збирання бонусів з незначного, в дуже важливе завдання.

З кожним наступним рівнем шлях гравця стає більше, через це зростає складність, платформи зникають швидше. Кожні 3–4 рівня з'являється жаба противник, яка хоче перешкодити гравцеві дійти до кінця.

У гравця є 5 життів. Як тільки гравець втратить 5–е життя, він програє і його рахунок фіксується. Гравець втрачає життя при падінні в воду, або при зіткненні з противником. Противник навчений бути перешкодою та шукає оптимальний шлях до героя, з розрахунком якості платформ, по яким цей шлях будується. Основні позначення можна побачити на рисунку (рис. 3.9).


Рисунок 3.9 – Основні позначення гри

Гравець розпочинає гру з невеликого навчання, в якому він навчиться стрибати та зрозуміє, що платформи зникають та стрибати на малу (платформу, що майже зникла) не варто, бо можна втратити життя і що для переходу на наступний рівень, необхідно стрибати до водяної лілії з квіткою. Після стрибку на водяну лілію з квіткою, через секунду рівень навчання завершиться, і буде завантажено перший рівень основної гри (рис. 3.10).

Рисунок 3.10 – Рівень-навчання

З прогресом по рівням, рівень стає дедалі більшим, а всі об'єкти зменшуються, з'являються бонуси та вороги, зустрічі з якими необхідно уникати (рис. 3.11).

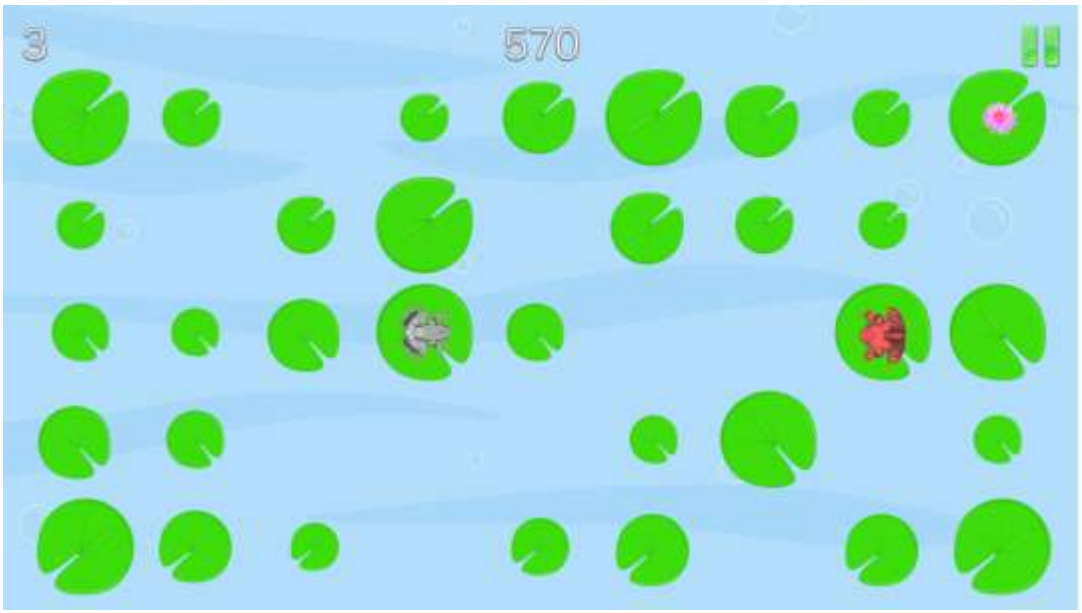


Рисунок 3.11 – Четвертий рівень

Задля уникнення проблем з відображенням об'єктів на екранах смартфонів та планшетів, які мають різний розмір та дозвіл экрана, рівень, елементи якого погано відображаються, відображається не повністю, а частково.

## 3.5 Програмна реалізація

У ході розробки гри «Frog» було вирішено чимало питань з приводу роботи ігрових механік певним та бажаним чином. І головним рішенням стало питання: «Як визначити, що платформи, на яку стрибає гравець немає?».

У грі гравець стрибає у будь-якому напрямку по горизонталі, чи вертикалі, крім границь екрану, незалежно від того, чи є платформа, чи її

немає. І гравець стрибає лише з платформи на платформу, він не може стрибнути через платформу, або по діагоналі.

Одним із рішень цього завдання є створення та активування будь-якого компонента GameObject (в даному випадку платформи) серед типу Collider2D: BoxCollider2D, CircleCollider2D, CapsuleCollider2D, PolygonCollider2D, чи CompositeCollider2D, коли платформа не відображається на екрані. Також необхідна умова, що при зіткненні коллайдер героя з коллайдером платформи, якої немає, відбувалася втрата життя, та початок рівня зі стартової позиції, якщо ще є життя, а якщо нема, то гра закінчувалася би (рис. 3.12).

```
public IEnumerator LilyAppear(GameObject lily)
{
    float randomIntervalToAppear = Random.Range(1.0f, 5.0f);
    yield return new WaitForSeconds(randomIntervalToAppear);
    lily.transform.localScale = new Vector2(LilyScale, LilyScale);
    _boxColider = lily.GetComponent<BoxCollider2D>();
    _boxColider.enabled = false;
    lily.GetComponent<SpriteRenderer>().enabled = true;
    StartCoroutine(LilyChangeSize(lily));
}
```

Рисунок 3.12 – Поява платформи на екрані з вимкненим коллайдером зіткнення з героєм

Цей метод являється корутином. Корутин – це метод в Unity3D, який дозволяє в одному потоці виконувати дії паралельно протягом певного часу. Вони обов'язково повинні мати тип «IEnumerator» та мати конструкцію «yield return». Корутини існують, тому що в цілому Unity3D працює в одному потоці. Цей корутин задає кожній платформі рандомний час до появи, від моменту створення гри, задає розмір об'єкта та отримує BoxCollider2D компонент, який ми вимикаємо, допоки життєвий цикл

платформи не буде закінчено та розпочинаємо корутин зі зміни розміру об'єкта з часом, який кожного етапу буде різним. (рис. 3.13).

```
public IEnumerator LilyChangeSize(GameObject lily)
{
    int i = 0;
    while (i < 4)
    {
        float randomIntervalToChange = Random.Range(1.0f, 5.0f);
        yield return new WaitForSeconds(randomIntervalToChange);
        lily.transform.localScale = lily.transform.localScale * 0.75f;
        i++;
        if (i == 4)
        {
            float randomIntervalToBeDissappered = Random.Range(1.0f, 5.0f);
            LilyDisappear(lily);
            yield return new WaitForSeconds(randomIntervalToBeDissappered);
            StartCoroutine(LilyAppear(lily));
        }
    }
}
```

Рисунок 3.13 – Поступова зміна розміру платформи з часом

Цей корутин має цикл while, тому що платформа мусить зменшуватися поступово, в три кроки. Розмір платформи зменшується з кожним кроком на 0,75 від минулого розміру і час кожної зміни в розмірі різний. Четвертий крок циклу while викличе метод LillyDisappear та корутин для нової появи цієї ж платформи через деякий час.

Цей метод отримує доступ до компоненту «Sprite Renderer», який відповідає за відображення об'єкта та вимикає його. А також змінює розмір колайдера до стартового (він також ставав меншим зі зменшенням розміру платформи), вмикає коллайдер, який використовується для відстеження ігрової логіки втрати життя.

## 3.6 Створення дизайну гри

Історично склалося, що в іграх дуже часто розробники експерементують з UX та UI і для різних платформ він виглядав у різний раз по різному. З появою сенсорних мобільних пристроїв з'явилися нові стандарти UX та UI у мобільних іграх. В залежності від проекту ці два аспекти дизайну мають різну функціональність та вигляд. Для гри «Frog» був обраний відносно простий стиль оформлення елементів меню, у стилі Material / Flat, який дозволяє легко, у декілька натисків перейти до будь-якого елемента: початок гри, налаштування зовнішнього вигляду героя, налаштування самої гри, досягнення, статистика, магазин речей (рис. 3.14).


Рисунок 3.14 – Головне меню гри ”Frog”

Головне меню динамічне. Вода тече, на ній з'являються хвильки, водяні лілії трохи рухаються, жабка анімована. У верхньому лівому кутку розташована кнопка переходу до меню, де гравець може змінити налаштування гри та мову, подивитися на свої досягнення, статистику та титри. Зверху посередині розташована назва гри. Зверху справа

розташована кнопка переходу до кастомізації головного героя, де гравець може подивитися на усі доступні йому зміни, дізнатися що необхідно зробити, щоб відкрити нові елементи, або подивитися ціну, за яку елемент можна відразу купити. Також гравець з цього меню може змінити анімацію стрибка, або зовнішній вигляд головного героя. По центру розташований герой у тому вигляді, у якому його хоче бачити гравець. Знизу розташована текстова підказка, як розпочати гру: для початку треба зробити жест swipe зліва-направо і жабка стрибне до лілії з квіткою. Камера зсунеться ліворуч, побудується рівень і розпочнеться гра.

При переході до пункту «Menu», гравець побачить спочатку налаштування гри, а також можливість перейти до вибору мови, досягнень, статистики та титрів (всі елементи стають видимими поступово, при переміщенні на наступний елемент) (рис. 3.15).


Рисунок 3.15 – Налаштування гри

На сторінці налаштувань користувач може змінить гучність звуків та музичного супровіду гри, використовуючи слайдер. Також гравець може

вибрати позицію для кнопки «Пауза», яка відображається під час гри. Ця опція існує, задля зручності користувача, тому що існують правші і левші. На платформі iOS існує облачний сервіс iCloud, який дозволяє берегти резервну копію даних гравця і гравець може використовувати цей сервіс. На платформі Android замість «iCloud», гравець побачить опцію «DropBox». Якщо гравець не зрозумів як керувати героєм, чи механіку гри, або він хоче дати пограти іншій людині вперше, він може ввімкнути туторіал знову. Також є опція стирання усього прогрессу, яка надає можливість розпочати гру з чистого аркушу.

У меню «Achievments» гравець може побачити усі доступні йому досягнення для відкривання (рис. 3.16).


Рисунок 3.17 – Меню досягнень гри

Усі досягення у списку гравець може добути, граючи у гру без жодних витрат реальних коштів. Досягення бувають простої, середньої та великої рівнями складності. Чим довше буде грати гравець, тим більше досягнень він здобуде, рано чи пізно. Досягнення також впливають на відкриття нових

кольорів для головного героя та елементів. Так як досягнень на даний момент 20 штук і усі не можливо розмістити на екрані одночасно, присутня можливість листати список досягнень. Про місцезнаходження у списку свідчить бігунець з правої сторони, який з'являється лише тоді, коли гравець намагається пролистати список.

# ВИСНОВКИ

У ході атестаційної роботи був проведен аналіз предметної області. Було виявлено існуючі на ринку аналоги планованої розробки. Було зроблено дослідження навчання з підкріпленням та методів його імплементації. На основі виконаного аналізу обрано два найоптимальніших методи навчання з підкріпленням, які було всебічно порівняно і досліджено на різних наборах даних.

Задля порівняння алгоритмів було виконано розробку гри “Frog”, яка потребує використання алгоритмів навчання з підкріпленням, щоб запропонувати користувачеві відчуття присутності реального супротивника. Аби порівняти алгоритми супротивник (агент) був навчений, використовуючи досліджувані алгоритми Q-learning та SARSA.

В процесі дослідження було застосовано різні конфігурації навчання та обрані найоптимальніші для кожного з алгоритмів на ігровому середовищі.

Після реалізації, дослідження та накопичення результатів навчання супротивника (агента) за допомогою наведених вище алгоритмів, було виконано глибинний аналіз та порівняння. У результаті аналізу було виявлено, що прогрес навчання алгоритмом Q-learning на початку стрімко росте, але після попадає в зону плато, де вже непомітно значних змін у поведінці агента. У той час алгоритм SARSA має більш пологий зліт, тобто розвиток є помітним раніше, ніж у випадку Q-learning, але він не є таким швидким, тобто для навчання йому потрібно більше часу. Коли навчання алгоритмом SARSA підходить до завершення, він показує приблизно однакові результати із Q-learning.

У результаті порівняння було виявлено, що алгоритм Q-learning краще використовувати при тривалому навчанні, в той час, як алгоритм SARSA краще використовувати при малій кількості епізодів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Джейсон Шрейер Зворотня сторона індустрії відеоігор. 2018р. с 55-60.

2. Тристан Донован Історія відеоігор 2019р. с 100-130

3. Внутриигровой баланс URL: https://www.epicgames.com/paragon/blog/ru/inside–game–balance (дата звернення: 05.10.20).

4. Game Development Tutorials URL: https://gamedevelopment.tutsplus.com/categories/unity (дата звернення: 22.09.20).

5. James Scott Nick Polson Artificial intelligence 2019р. с 273-280.

6. Georgios N. Yannakakis, Togelius J. Artificial Intelligence and Games. New York; London: New York University Press, 2018, 337 p.

7. A Beginner's Guide to Deep Reinforcement Learning URL: https://skymind.ai/wiki/deep-reinforcement-learning(дата звернення: 03.05.19).

8. Simple Reinforcement Learning:Qlearning URL: https://towardsdatascience.com/simple-reinforcement-learning-q-learning (дата звернення: 10.05.19).

9. Reinforcement Learning Demystified: Markov Decision Processes URL: https://towardsdatascience.com/reinforcement-learning-demystified-markov-decision-processes (дата звернення: 20.08.20).

10. Oxland K.J. Gameplay and design Addison Wesley, 2004. 368 с.

11. Ричард С. Саттон Эндрю Дж. Барто. Навчання з підкріпленням. 2020р. с 367-400

12. Stephen Finley. Artificial Intelligence and Machine Learning for Business 2018р. с 130-140

13. Аарон Курвіль, Йошуа Бенжіо, Ян Ґудфелоу. Глибоке навчання. 2027р. с 30-70

14. Jerome Harold Friedman, Robert Tibshirani, and Trevor Guest. The

Elements of Statistical Learning 2017p. c 65-70

15. Oliver Theobald. Machine Learning for Absolute Beginners: A Plain English Introduction 2017p. c 400-410

16. Пітер Норвіг, Стюарт Расселл. Штучний інтелект: сучасний підхід 2020р. с 100-124

17. A Beginners Guide to Q-learning URL: https://towardsdatascience.com (дата звернення: 03.08.20)

18. An introduction to Q-learning: Reinforcement Learning URL: https://blog.floydhub.com ()

19. Real-World Applications of Q-Learning: Gentle Introduction URL: https://perfectial.com/blog/q-learning-applications (дата звернення: 05.09.20)

20. SARSA Reinforcement Learning URL: https://www.geeksforgeeks.org (дата звернення: 10.09.20)

21. Introduction to Reinforcement Learning (Coding SARSA) URL: https://medium.com/swlh/introduction-to-reinforcement-learning (дата звернення: 13.09.20)

22. Q-Learning, Expected Sarsa and comparison of TD learning algorithms URL: https://medium.com/analytics-vidhya (дата звернення: 13.10.20)

23. On-Policy v/s Off-Policy Learning URL: https://towardsdatascience.com/on-policy-v-s-off-policy-learning (дата звернення: 20.10.20)

24. On-Policy VS Off-Policy in Reinforcement Learning URL: https://leimao.github.io/blog/RL-On-Policy-VS-Off-Policy (дата звернення: 10.11.20)

25. Nechiporenko A.S., Gubarenko E.V., Gubarenko M.S. (2019) Authentication of users of mobile devices by their motor reactions. Telecommunications and Radio Engineering, 78 (11). pp. 987-1003. DOI: 10.1615/TelecomRadEng.v78.i11.60